1.1 SIGNAL DESCRIPTIONS -OF 8086

The microprocessor 8086 is a 16-bit CPU available in three clock rates, i.e. 5, 8 and 10MHz, packaged in a 40 pin CERDIP or plastic package. The 8086 operates in single processor or multiprocessor configurations to achieve high performance. The pin configuration is shown in Fig. 1.1. Some of the pins serve a particular function in minimum mode (single processor mode) and others function in maximum mode (multiprocessor mode) configuration.

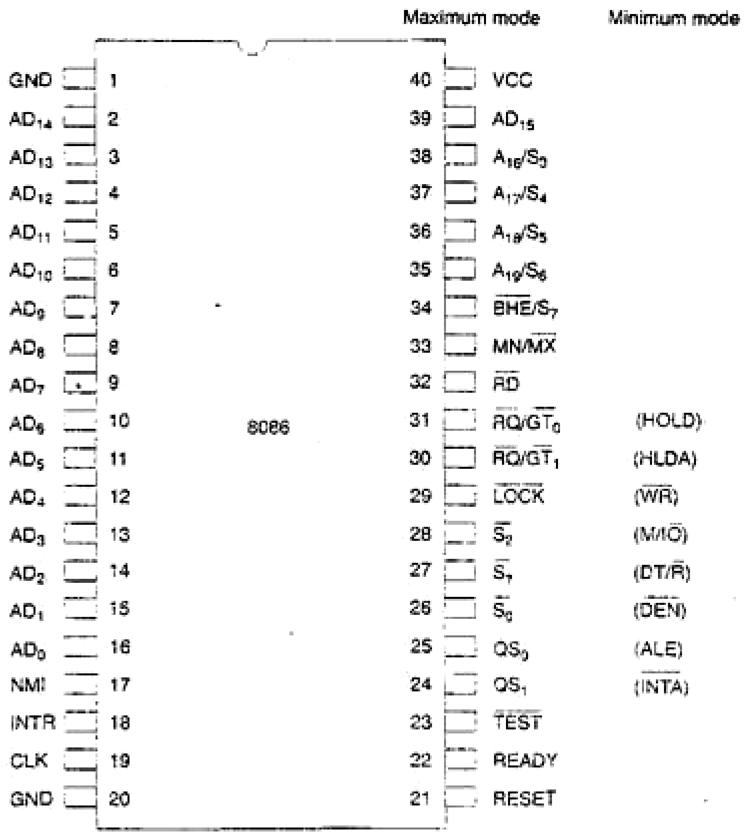


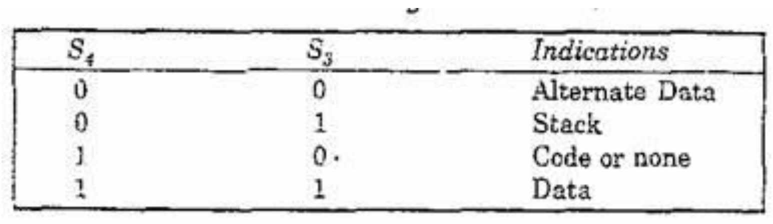
Fig. 1.1 *Pin Configuration of* 8086

The 8086 signals can be categorized in three groups. The first are the signals having common functions in minimum as well as maximum mode, the second are the signals which have special functions for minimum mode and the third are the signals having special functions for maximum mode.

The following signal descriptions are common for both the minimum and maximum modes.

AD15 - AD0 These are the time multiplexed memory I/O address and data lines. Address remains on the lines during the first clock cycle of instruction cycles, while the data is available on the data bus during later clock cycles.

A19/S6, A16/S5, A17/S4, A16/S3 These are the time multiplexed address and status lines. During the first clock cycle of instruction cycles, these are the most significant address lines for memory operations. During I/O operations, these lines are low. During memory or I/O operations, status information is available on those lines during later clock cycles. The status of the interrupt enable flag bit (displayed on S5) is updated at the beginning of each clock cycle. The S4 and S3 combinedly indicate which segment register is presently being used for memory accesses as shown in Table 1.1. The status line S3 is always low (logical). The address bits are separated from the status bits using latches controlled by the ALE signal.



S_4	S_3	*Indications*
0	0	Alternate Data
0	1	Stack
1	0	Code or none
1	1	Data

Table 1 .1 Bus High Enable/status

$\overline{\text{BHE}}$/S7 - Bus High Enable/Status The bus high enable signal is used to indicate the transfer of data over the higher order (D15—D8) data bus as shown in Table 1.2. It goes low for the data transfers over D0—D7 and is used to derive chip selects of odd address memory bank or peripherals. BHE is low during T1 for read, write and interrupt acknowledge cycles, whenever a byte is to be transferred on the higher byte of the data bus.

$\overline{\text{RD}}$ - Read Read signal when low, indicates the peripherals that the processor is performing a memory or I/O read operation. RD is active low output signal.

READY This is the acknowledgement from the slow devices or memory that they have completed the data transfer. READY is an input active high.

INTR - lnterrupt Request This is a level triggered input. This is sampled during the last clock cycle of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle. This can be internally masked by resetting the interrupt enable flag. This signal is active high and internally synchronized.

$\overline{\text{TEST}}$ This input is examined by a 'WAIT' instruction. If the TEST input goes low, execution will continue, else, the processor remains in an idle state. The input is synchronized internally during each clock cycle on leading edge of clock.

NMI - Non-maskable interrupt This is an edge-triggered input which causes a Type2 interrupt. The NMI is not maskable internally by software. A transition from low to high initiates the interrupt response at the end of the current instruction. This input is internally synchronized.

RESET This input causes the processor to terminate the current activity and start execution from FFFFOH. The signal is active high and must be active for at least four clock cycles. It restarts execution when the RESET returns low. RESET is also internally synchronized.

CLK - Clock Input The clock input provides the basic timing for processor operation and bus control activity. It is a asymmetric square wave with 38% duty cycle. The range of frequency for different 8086 versions is from 5MHz to 10MHz.

Vcc +5V power supply for the operation of the internal circuit.

GND ground for the internal circuit.

MN/$\overline{\text{MX}}$ The logic level at this pin decides whether the processor is to operate in either ‘minimum (single processor) or maximum (multiprocessor) mode.

The following pin functions are for the minimum mode operation of 8086.

M/$\overline{\text{IO}}$ -Memory/IO This is a status line logically equivalent to S2 in maximum mode. When it is low, it indicates the CPU is having an I/O operation, and when it is high, it indicates that the CPU is having a memory operation.

$\overline{\text{INTA}}$ - Interrupt Acknowledge This signal is used as a read strobe for interrupt acknowledge cycles. In other words, when it goes low, it means that the processor has accepted the interrupt.

ALE - Address Latch Enable This output signal indicates the availability of the valid address on the address/data lines, and is connected to latch enable input of latches. This signal is active high.

DT/$\overline{\text{R}}$ - Data Transmit/Receive This output is used to decide the direction of data flow through the transreceivers (bidirectional buffers). When the processor sends out data, this signal is high and when the processor is receiving data, this signal is low.

$\overline{\text{DEN}}$-Data Enable This signal indicates the availability of valid data over the address/data lines. It is used to enable the transreceivers (bidirectional buffers) to separate the data from the multiplexed address/data signal.

HOLD, HLDA - Hold/HoId Acknowledge When the HOLD line goes high, it indicates to the processor that another master is requesting the bus access. The processor, after receiving the HOLD request, issues the hold acknowledge signal on

HLDA pin, in the middle of the next clock cycle after completing the current bus (instruction) cycle. At the same time, the processor floats the local bus and control lines. When the processor detects the HOLD line low, it lowers the HLDA signal. HOLD is an asynchronous input, and it should be externally synchronized.

1.2 PHYSICAL MEMORY ORGANISATION

In an 8086 based system, the 1Mbytes memory is physically organized as odd bank and even bank, each of 5l2 Kbytes, addressed in parallel by the processor. Byte data with even address is transferred on D7-D0, while the byte data with odd address is transferred on D15-D8 bus lines. The processor provides two enable signals, BHE and A0 for selection of either even or odd or both the banks. The instruction stream is fetched from memory as words and is addressed internally by the processor as necessary. In other words, if the processor fetches a word (consecutive two bytes) from memory, there are different possibilities, like:

1. Both the bytes may be data operands.
2. Both the bytes may contain opcode bits.
3. One of the bytes may be opcode while the other may be data.

All the above possibilities are taken care of by the internal decoder circuit of the microprocessor. The opcodes and operands are identified by the internal decoder circuit which further derives the signals those act as input to the timing and control unit. The timing and control unit then derives all the signals required for execution of the instruction.

In referring word data, the BIU requires one or two memory cycles, depending upon whether the starting byte is located at an even or odd address. It is always better to locate the word data at an even address. To read or write a complete word from/to memory, if it is located at an even address, only one read or write cycle is required. If the word is located at an odd address, the first read or write cycle is required for accessing the lower byte while the second one is required for accessing the upper byte. Thus two bus cycles are required, if a word is located at an odd address. It should be kept in mind that while initializing the structure like stack they should be initialized at an even address for efficient operation.

Certain locations in memory are reserved for specific CPU operations. The locations from FFFFOH to FFFFFH are reserved for operations including jump to initialization programme and I/O-processor initialization. The locations 00000H to 003FFH are reserved for interrupt vector table. The interrupt structure provides space for a total of 256 interrupt vectors. The vectors, i.e. CS and IP for each interrupt routine requires 4 bytes for storing it in the interrupt vector table. Hence 256 types of interrupt require 256 x 4 = 03FFH (1Kbyte) locations for the complete interrupt vector table.

1.3 I/O ADDRESSING CAPABILITY

The 8086/8088 processor can address up to 64K I/O byte registers or 32K word registers. The limitation is that the address of an I/O device must not be greater than 16 bits in size, this means that a maximum number of 2^{16}, i.e. 64 Kbyte I/O devices may be accessed by the CPU. The I/O address appears on the address lines A0 to A15 for one clock cycle (T1). It may then be latched using the ALE signal. The upper address lines (A16 — A19) are at logic 0 level during the IO operations. The 16-bit register DX is used as 16-bit I/O address pointer, with full capability to address up to 64K devices. In this case, the IO ports are addressed in the same manner as memory locations in the based addressing mode using BX. In memory mapped I/O interfacing, the I/O device addresses are treated as memory locations in page 0, i.e. segment address 0000H. Even addressed bytes are transferred on D7—D0 and odd addressed bytes are transferred on D8— D15 lines. While designing any 8 bit IO system around 8086, care must be taken that all the byte registers in the system should be addressed even.

1.4 I/O Techniques

This section describes the basic input and output techniques used by microcomputers to transfer data between the microcomputer and external devices. The general characteristics of I/O are described. One communicates with a microcomputer system via the I/O devices interfaced to it. The user can enter programs and data using the keyboard on a terminal and execute the programs to obtain results. Therefore, the I/O devices connected to a microcomputer system provide an efficient means of communication between the computer and the outside world. These I/O devices are commonly called peripherals and include keyboards, CRT displays, printers, and disks.
The characteristics in the I/O devices are normally different from those of the microcomputer. For example, the speed of operation of the peripherals is usually slower compared to the microcomputer, and the word length of the microcomputer may be different from the data format of the peripheral device. To make the characteristics of the I/O devices compatible with those of the microcomputer, interface hardware circuitry between the microcomputer and I/O devices is necessary. In a typical microcomputer system, the user gets involved with two types of I/O devices: physical I/O and logical I/O. When the microcomputer has no operating system, the user must work directly with physical I/O devices and perform detailed I/O design. There are three ways of transferring data between the microcomputer and a physical I/O device:

1.4.1 Simple I/O

When you need to get digital data from some simple switch, such as a thermostat into a microprocessor, all you have to do is connect the switch to an input port line and read the port. The thermostat data is always present and ready, so you can read it at any time. Likewise, when you need to output data to a simple display device such as an LED, all you have to do is to connect the input of the LED buffer on an output port pin and output the logic level required to turn on the light. The LED is always there and ready, so you can send data to it at any time. The timing waveform in Figure 1.2a represents this situation. The crossed lines on the waveform represent the time at which a new data byte becomes valid on the output lines of the port. The absence of other waveforms indicates that this output operation is not directly dependent on any other signals.

1.4.2 Conditional I/O

In conditional I/O, the microprocessor outputs data to an external device via handshaking. Data transfer occurs by the exchanging of control signals between the microprocessor and an external device. The microprocessor inputs the status of the external device to determine whether the device is ready for data transfer. Data transfer takes place when the device is ready. According to the way of exchanging the handshake signals, conditional I/O can be classified to; simple strobe I/O, single handshake I/O, double handshake I/O.

Simple Strobe I/O

In many applications valid data is only present on an external device at a certain time and it must be read in at that time. When a key is pressed on a keyboard, circuitry on the keyboard sends out the ASCII code for the pressed key on parallel data lines. The keyboard circuitry then sends out a strobe signal on another line to indicate that valid data is present on the eight data lines. You can connect this strobe line to an input port line and poll it to determine when you can input valid data from the keyboard. Another alternative is to connect the strobe line to an interrupt input on the processor and have an interrupt service routine read in the data when the processor receives an interrupt. The point here is that this transfer is time-dependent. You can only read in data when a strobe pulse tells you that the data is valid.

Figure 1.2b shows the timing waveforms which represent this type of operation. The sending device, such as a keyboard, outputs parallel data on the data lines and then outputs an STB signal to let you know that valid data is present.

For low rates of data transfer, such as from a keyboard to a microprocessor, a simple strobe transfer works well, However, for high-speed data transfer this method does not work because there is no signal which tells the sending device when it is safe to send the next data byte. In other words the sending system might send data faster than the receiving system could read them. To prevent this problem a handshake data transfer scheme is used.

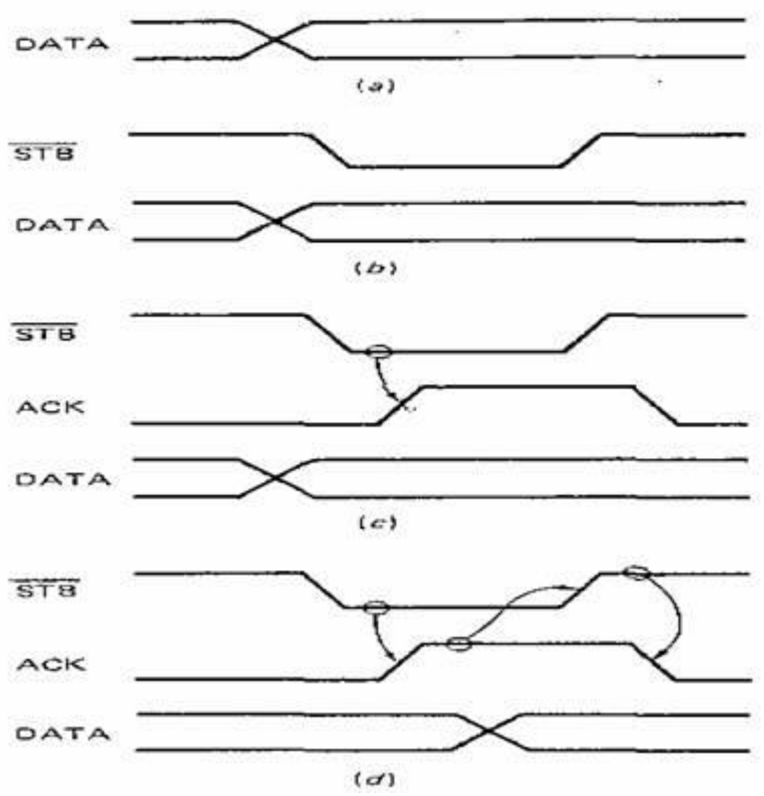


Figure 2.1: Data Transfer. *(a) Simple I/O. (b) Simple Strobe I/O. (c) Single Handshake I/O. (d) Double Handshake I/O*

SINGLE HANDSHAKE

Figure 1.2c shows some example timing waveforms for a handshake data transfer from a peripheral device to microprocessor. The peripheral outputs some data and sends an STB signal to the microprocessor. The microprocessor detects the asserted STB signal on a polled or interrupt basis, and reads in the data. The microprocessor then sends an acknowledge signal ACK, to the peripheral to indicate that the peripheral can send the next byte of data. From the viewpoint of the microprocessor, this operation Is referred to as a handshake or strobed input.

These same waveforms might represent a handshake output from a microprocessor to a parallel printer, this case the microprocessor outputs a character to the printer and asserts an STB signal to the printer to tell the printer, ‘Here is a character for you.” When the printer is ready, it answers back with the ACK signal to tell the microprocessor. “I got that one, send me another.” The point of this handshake scheme is that the sending device or system cannot send the next data unit until the receiving device or system indicates with ACK signal that it is ready to receive the next byte.

DOUBLE HANDSHAKE DATA TRANSFER

For data transfers where even more coordination is required between the sending system and the receiving system, a double handshake is used. Figure 1.2d, shows some example waveforms for a double handshake input from a peripheral to a microprocessor. Perhaps it will help you to follow these waveforms by thinking of them as a conversation between two people. In these waveforms each signal edge has meaning. The sending device asserts its STB line low to ask. “Are you ready? The receiving system raises its ACK line high to say, “I'm ready.” The peripheral device then sends the data and raises Its STB line high to say. “Here is some valid data for you.’ After it has read in the data the receiving system drops its ACK line low to say, “I have the data, thank you, and I await your request to send the next unit of data,”

For a handshake output of this type, from a microprocessor to a peripheral, the waveforms are the same but the microprocessor sends the STB signal and data, and the peripheral sends the ACK signal.

For handshake data transfer, a microprocessor can determine when it is time to send the next data unit on a polled or on an interrupt basis. The interrupt approach makes better use of the processor’s time.

1.4.3 Direct memory access (DMA)

Direct Memory Access (DMA) is a technique that transfers data between a microcomputer’s memory and an I/O device without involving the microprocessor. DMA is widely used in transferring large blocks of data between a peripheral device and the microcomputer’s memory. The DMA technique uses a DMA controller chip for the data-transfer operation. The main functions of a typical DMA controller are summarized as follows:

1. The I/O devices request DMA operation via the DMA request lines of the controller chip.

2. The controller chip activates the microprocessor HOLD pin, requesting the CPU to release the bus.

3. The processor sends HLDA (hold acknowledge) back to the DMA controller, indicating that the bus is disabled. The DMA controller places the current value of its internal registers, such as the address register and counter, on the system bus and sends a DMA acknowledge to the peripheral device. The DMA controller completes the DMA transfer and releases the buses.